**第12章 文件**

目录

[**一、** **缓冲文件系统的工作原理** 1](#_Toc391815831)

[**二、** **打开文件** 1](#_Toc391815832)

[**1.** **文件指针** 1](#_Toc391815833)

[**2.** **文件处理步骤** 2](#_Toc391815834)

[**3.** **打开文件** 2](#_Toc391815835)

[**4.** **常用来打开文件的写法** 2](#_Toc391815836)

[**5.** **打开文件的方式** 2](#_Toc391815837)

[**三、** **关闭文件** 4](#_Toc391815838)

[**四、** **文件读写函数** 4](#_Toc391815839)

[**1.** **字符方式文件读写函数** 4](#_Toc391815840)

[**2.** **字符串方式文件读写函数** 6](#_Toc391815841)

[**3.** **格式化方式文件读写函数** 8](#_Toc391815842)

[**4.** **数据块方式文件读写函数** 11](#_Toc391815843)

1. **缓冲文件系统的工作原理**

程序要操作磁盘文件的数据，必须要借助缓冲区，可以大大提高文件操作的速度。

缓冲文件系统的工作原理如下图所示：

|  |
| --- |
| C程序 |
|  |
|  |
|  |
|  |

内存缓冲区

|  |
| --- |
|  |
|  |
|  |
|  |

磁盘

的

操作系统

自动完成

程序控制

**fp**

**512B**

1. **打开文件**
2. **文件指针**

C语言中文件操作统一以文件指针方式实现。

定义**文件指针**的格式为： FILE \*fp;

文件指针是特殊指针，指向的是文件类型结构，它是多项信息的综合体。

文件操作具有顺序性的特点，前一个数据取出后，下一次将顺序取后一个数据，这样的操作由操作系统在文件读写时自动完成。

1. **文件处理步骤**

C语言编写文件操作的程序要遵循如下步骤：

1. 定义文件指针
2. 打开文件：文件指针指向磁盘文件缓冲区
3. 文件处理：文件读写操作
4. 关闭文件
5. **打开文件**

fopen()函数实现打开文件，调用形式为：

fopen(“文件名”, “文件打开方式”);

注：（1）函数返回值

（2）文件名：一般指定文件的路径，若不写则默认与应用程序的当前路径相同。若为绝对路径，则定位用的斜杠“\”需要用“\\”。

例：

|  |
| --- |
| FILE \*fp;  fp = fopen(“myfile.txt”, “r”); // fp = fopen(“D:\\myfile.txt”, “r”); |

1. **常用来打开文件的写法**

|  |
| --- |
| /\*为保证打开文件操作的可靠性，做下面的判断，确保文件正常打开后再进行读写\*/  if((fp=fopen("test.txt","r"))==NULL)  {  printf("cannot open a file!\n");  exit(0);  }  /\*  exit()函数  关闭已打开的所有文件，结束程序运行，返回操作系统，并将“程序状态值”返回给操作系统。当值为０时，表示程序正常退出；非０值时，表示程序出错退出。  \*/ |

1. **打开文件的方式**

**（1）基本参数用法**

|  |  |
| --- | --- |
| 参数 | 参数的意义 |
| “r” | 表示为读而打开文本文件。  注意：打开不存在文件或不允许读的文件会出错。  不会自创建文件。 |
| “w” | 表示为写而打开文本文件。  注意：若不存在，会创建文件。  若存在，则从文件首部开始写，原有内容将消失。 |
| “a” | 表示向文件末尾追加数据。  注意：若不存在，会创建文件。  若存在，则从文件末尾开始写。 |
| “b” | 表示二进制 |
| “+” | 表示读和写、写和读 |

**（2）参数的组合用法**

|  |  |
| --- | --- |
| 参数 | 参数的意义 |
| “rb” | 表示为读而打开二进制文件。其余类似“r” |
| “wb” | 表示为写而打开二进制文件。其余类似“w” |
| “ab” | 表示为添加而打开二进制文件。其余类似“w” |

|  |  |
| --- | --- |
| 参数 | 参数的意义 |
| “r+” | 表示为读和写而打开文本文件。  注意：1. 文件已存在  2. 读和写都是从文件首部开始  3. 写新数据时，要覆盖新数据 |
| “w+” | 表示为写和读而打开文本文件。  注意：1. 文件存在，则原内容消失  2. 文件写完后，可以从文件首部开始读 |
| “a+” | 表示为添加而打开文本文件。  注意：1. 添加数据后，从文件首部开始读 |

|  |  |
| --- | --- |
| 参数 | 参数的意义 |
| “rb+” | 表示为读和写而打开二进制文件。  注意：不一定从文件首部开始写 |
| “wb+” | 表示为写和读而打开二进制文件。  注意：可以指定位置操作 |
| “ab+” | 表示为添加而打开二进制文件。  注意：添加数据后，可以指定位置读 |

1. **关闭文件**

编程时养成文件使用结束后及时关闭文件的习惯，一则确保数据完整写入文件，二则及时释放不用的文件缓冲区单元。

关闭文件通过调用函数**fclose(文件指针)**实现，一般格式为：

|  |
| --- |
| fclose(fp); //fp是文件指针 |

通常使用条件判断格式：

|  |
| --- |
| /\*文件操作完成后，应及时关闭。正常关闭文件返回值为0，否则关闭失败。\*/  if(fclose(fp))  {  printf(Failure to close the file!\n");  exit(0);  } |

1. **文件读写函数**

C语言提供了一系列文件的读写操作函数，常用的函数如下：

1. **字符方式文件读写函数**

字符方式文件读写函数fgetc() 和 fputc() 。

1. **fgetc()函数从fp所指示的磁盘文件读入一个字符。**

格式为：**ch = fgetc(fp);**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  void main()  {  FILE \*fp;  char ch;  /\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  "r"为读而打开文本文件  1. 文件不存在，则不会创建该文件；  2. 若存在，but是不允许读的文件，会出错  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/  /\*为保证打开文件操作的可靠性，做下面的判断，确保文件正常打开后再进行读写\*/  if((fp=fopen("test.txt","r"))==NULL)  {  printf("cannot open a file!\n");  exit(0);  }  /\*从指定的文件输出所有字符\*/  /\* 以EOF作为文件结束标志，其值通常为-1 (必须是文本文件) \*/  while((ch=fgetc(fp))!=EOF)  {  printf("%c",ch); //putchar(ch);  }  printf("\n");  /\*文件操作完成后，应及时关闭。正常关闭文件返回值为0，否则关闭失败。\*/  if(fclose(fp))  {  printf("cannot close the file!\n");  exit(0);  }  }  /\* rewind(fp);  ch=fgetc(fp);  printf("%c",ch);  \*/ |

1. **fputc()函数把一个字符写到fp所指示的磁盘文件上。**

**格式为：ch = fputc(fp);**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  void main()  {  FILE \*fp;  char ch;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  "w"为写而打开文本文件  1. 文件不存在，则会创建该文件；  2. 若存在，则从文件起始位置开始写，原有内容将全部消失  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*为保证打开文件操作的可靠性，做下面的判断，确保文件正常打开后再进行读写\*/  if((fp=fopen("test.txt","w"))==NULL)  {  printf("cannot open a file!\n");  exit(0);  }  ch=getchar(); //输入一个字符  fputc(ch,fp); //fputc(ch,fp)把一个字符ch写到fp所指示的磁盘文件上  /\*文件操作完成后，应及时关闭。正常关闭文件返回值为0，否则关闭失败。\*/  if(fclose(fp))  {  printf("cannot close the file!\n");  exit(0);  }  } |

1. **字符串方式文件读写函数**

字符串方式文件读写函数fgets() 和fputs() 。

1. **fputs()函数向指定的文本文件写入一个字符串。**

**格式为：fputs(string, fp);**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  void main()  {  FILE \*fp;  char \*str="This is a short string.";  /\*为保证打开文件操作的可靠性，做下面的判断，确保文件正常打开后再进行读写\*/  if((fp=fopen("test.txt","w"))==NULL)  {  printf("cannot open a file!\n");  exit(0);  }  /\* fputs(str,fp)向指定的文本文件写入一个字符串 \*/  fputs(str,fp);  /\*文件操作完成后，应及时关闭。正常关闭文件返回值为0，否则关闭失败。\*/  if(fclose(fp))  {  printf("cannot close the file!\n");  exit(0);  }  }  /\*  fseek(fp,5,0); //定位后，写入字符串  \*/  /\*  fseek(fp,0,2);//重新定位  fputs("append",fp); //写入常量字符串  \*/ |

1. **fgets()函数从文本文件中读取字符串。**

格式为：**fgets(string, n, fp);**

注：n为指定读入的字符个数，调用时最多读入n-1个字符

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  void main()  {  FILE \*fp;  char \*str="This is a short string.";  char s[100];  /\* "w+":建立一个新文件进行写，随后可以从头开始读。  若文件存在，则原内容全部消失。  \*/  /\*为保证打开文件操作的可靠性，做下面的判断，确保文件正常打开后再进行读写\*/  if((fp=fopen("test.txt","w+"))==NULL)  {  printf("cannot open a file!\n");  exit(0);  }  fputs(str,fp); //写入字符串  /\* fgets(str,n,fp) 从指定的文本文件读取字符串,只读出字符串的前n-1个字符\*/  fseek(fp,0,0); //定位指针到文件首部  fgets(s, strlen(str)+1, fp);  /\*  fseek(fp,0,2);  fputs(s,fp);  puts(s); //测试s中存贮的是fgets函数读取的字符串  \*/  /\*文件操作完成后，应及时关闭。正常关闭文件返回值为0，否则关闭失败。\*/  if(fclose(fp))  {  printf("cannot close the file!\n");  exit(0);  }  } |

1. **格式化方式文件读写函数**

格式化方式文件读写函数fscanf() 和fprintf() 。

1. **从文件按给定的控制格式读取数据。**

格式为：**fscanf(文件指针, 格式字符串, 输入表);**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  void main()  {  FILE \*fp;  int num,score;  char str[10];    /\*为保证打开文件操作的可靠性，做下面的判断，确保文件正常打开后再进行读写\*/  if((fp=fopen("test.txt","r"))==NULL)  {  printf("cannot open a file!\n");  exit(0);  }  /\*输出字段名，即表头\*/  printf("num name score\n");  /\*判断是否到了文件尾，若没有则执行循环体，从文件中按照指定的格式读取数据\*/  while(!feof(fp))  {  fscanf(fp,"%d %s %d",&num,str,&score);  printf("%-5d%-10s%-6d\n",num,str,score);  }  /\*文件操作完成后，应及时关闭。正常关闭文件返回值为0，否则关闭失败。\*/  if(fclose(fp))  {  printf("cannot close the file!\n");  exit(0);  }  } |

1. **按照给定的控制格式向文件中写入数据。**

格式为：**fprintf(文件指针, 格式字符串, 输入表);**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  void main()  {  FILE \*fp;  int num,score;  char str[10];  int num1,math;  char ch[10];    /\*为保证打开文件操作的可靠性，做下面的判断，确保文件正常打开后再进行读写\*/  if((fp=fopen("test.txt","w+"))==NULL)  {  printf("cannot open a file!\n");  exit(0);  }    /\* fprintf(文件指针,格式控制,输出表)  1. 若是写文件，则将按照格式写入数据到文件  2. 若是读文件，则从文件中按给定的格式读取数据保存到变量  \*/  fprintf(fp,"%d %s %d",1,"weitao",98);  /\* 重新定位指针位置为文件首部 \*/  rewind(fp);  /\*输出字段名，即表头\*/  printf("num name score\n");  /\*判断是否到了文件尾，若没有则执行循环体，从文件中按照指定的格式读取数据\*/  while(!feof(fp))  {  //fscanf(文件指针,格式控制，输入表)从文件中按照指定的格式读取数据  fscanf(fp,"%d %s %d",&num,str,&score);  printf("%-5d%-10s%-6d\n",num,str,score);  }  /\*文件操作完成后，应及时关闭。正常关闭文件返回值为0，否则关闭失败。\*/  if(fclose(fp))  {  printf("cannot close the file!\n");  exit(0);  }  }  /\*  int n=10;  char string[10];  int math=99;    fprintf(fp,"%d %s %d\n",n,string,math);  \*/ |

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #define MAX 100  struct student  {  int num;  char name[10];  int average;  };  void main()  {  FILE \*fp;  struct student stu[MAX];  int N,i;  int k=0;  /\*为保证打开文件操作的可靠性，做下面的判断，确保文件正常打开后再进行读写\*/  if((fp=fopen("test.txt","w+"))==NULL)  {  printf("cannot open a file!\n");  exit(0);  }  printf("请输入学生的个数：");  scanf("%d",&N);  //输入学生信息  for(i=0;i<N;i++)  {  printf("请输入第%d个学生的信息：",i+1);  scanf("%d%s%d",&stu[i].num,stu[i].name,&stu[i].average);  }  for(k=0;k<N;k++)  {  fprintf(fp,"%10d %10s %10d\n",stu[k].num,stu[k].name,stu[k].average);  }  fclose(fp);  } |

1. **数据块方式文件读写函数**

**数据块方式文件读写函数fread() 和fwrite()**

1. **fread(buffer, size,count,fp);**
2. **fwrite(buffer, size,count,fp);**